

REMARKS

Claims 1-33 are pending in the present application and stand rejected. The Examiner's reconsideration is respectfully requested in view of the following remarks.

Claims 1-33 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Torii (U.S. Patent No. 6,122,712) in view of Steckermeier et al ("Using Locality Information in Userlevel Scheduling) (hereinafter "Steckermeier"). The rejection is respectfully traversed.

The Examiner maintains that col.3, lines 47-51 of Torii teaches or suggests "storing in a first data structure thread ids for at least some of the threads associated with a context switch performed by the operating system, each of the thread ids uniquely identifying one of the threads," as claimed in claim 1. Applicants respectfully disagree.

The recited portion of claim 1 begins with the step of *storing* thread ids in a first data structure. The recited portion of Torii (i.e., col. 3, lines 47-51) does not teach or suggest a *storing* step. In particular, the recited portion of Torii discloses a "thread management unit" for *allocating* thread identifiers to threads, and *notifying* a "thread identifier register" of the thread identifiers. The steps of *allocating* and *notifying* thread identifiers, as disclosed in Torii, do not teach or suggested the *storing* of thread ids in a first data structure, as essentially claimed in claim 1.

The Examiner argues that "the *thread management unit* acts as a database containing information pertaining to all system threads." (Paper no. 11052004, p. 13). However, as shown in detail above, any argument that the thread management unit of Torii acts as a *database* is entirely speculative. The recited portion of Torii simply discloses that a thread management unit allocates and notifies. No teaching or suggestion is made in the

recited portion of Torii that the thread management unit has *any* storage capabilities for thread identifiers.

The Examiner further argues: “A context switch, as it relates to multithreading, is simply a processor’s changing execution from one thread to another. As the thread management unit stores information relating to all system threads, it follows that the thread management unit would store information relating to threads.” (Paper no. 1-1052004, p. 13). Applicants respectfully submit that the prior sentence is not precise. Context switching involves an explicit *process*. A primary purpose of context switching in a multithreading environment is to switch from one running thread to another (e.g., saving a current thread, flushing the memory, and loading the new thread). Nothing in the recited portion of Torii shows the execution of a process to perform a context switch, or even a need to perform a context switch. In particular, the recited portion of Torii describes assigning thread identifiers as threads are generated, and no threads are even running. It does not make sense to perform a potentially computationally-intensive context switch when thread identifiers are simply assigned as threads are generated. Any argument to the contrary is not only speculative (i.e., arguing that Torii teaches or suggests a context switch when no such teaching or suggestion can be found in the reference), but also an improper interpretation of the science as understood by one skilled in the art.

For purposes of the remaining arguments, Applicants will assume, *arguendo*, that the “thread management unit,” as disclosed in Torii, renders obvious the “first data structure” for storing thread ids, as essentially claimed in claim 1.

The Examiner maintains that col. 4, lines 25-30 of Torii teaches or suggests “storing in a second data structure a plurality of entries for a plurality of groups of

contiguous cache lines,” as claimed in claim 1. The recited portion of Torii (i.e., col. 4, lines 25-30) recites: “The cache coherency maintenance protocol sequencer 14 *controls protocol* between a local cache memory and another cache memory or the main memory based on the information transmitted from the comparator 13c.” Once again, the recited portion of claim 1 includes a step of *storing* a plurality of entries in a second data structure. A sequencer for *controlling protocol* between memories is not even remotely related to *storing* a plurality of entries. Nothing in the recited portion of Torii teaches or suggests a step of *storing* a plurality of entries.

The Examiner maintains that col. 4, lines 31-48 of Torii teaches or suggests “each of the plurality of entries arranged such that a thread id in the first data structure *is capable of being associated* with at least one of the contiguous cache lines in at least one of the plurality of groups of contiguous cache lines, the thread identified by the thread id *having accessed* the at least one of the contiguous cache lines in the at least one of the plurality of groups of contiguous cache lines,” as claimed in claim 1.

The recited portion of Torii (i.e., col. 4, col. 31-48) describes Figure 4 of Torii, which is a block diagram of *a cache memory* as shown in Figure 2 (i.e., units 3a to 3d). In particular, the recited portion of Torii describes four statuses for a cache line. The Examiner elaborates that “[t]he cache...stores data relating to the thread which last accessed it,” and that “locating cache lines that were accessed by the thread allows the cache coherency controller to identify other cache lines that may be accessed by the thread to reduce the number of miss penalties.” (Paper no. 11052004, p. 13).

Notwithstanding that both of the Examiner’s elaborations are entirely speculative as neither statement is supported by the recited portion of Torii, the Examiner’s argument

does not precisely address the claim language of claim 1. The Examiner basically ignores that claim 1 claims that “*a thread id* in the first data structure is capable of being associated with at least one of the contiguous cache lines.” The recited portion of Torii does not even access the thread management unit, which the Examiner previously argued is a database that stores the thread identifiers. Further, the Examiner at one point seems to argue that the “coherency maintenance protocol sequencer” as teaching or suggesting the “second data structure,” and now, seems to argue the “cache memory” as teaching or suggesting the “second data structure.” This apparent inconsistency needs to be clarified.

The Office Action cites col. 4, lines 13-18 and col. 4, line 66 to col. 5, line 7 of Torii as teaching or suggesting “mining for patterns in the plurality of entries in the second data structure to locate multiples of a same thread id that repeat with respect to at least two of the plurality of groups of contiguous cache lines,” as claimed in claim 1.

It is unclear which part of Torii teaches or suggests “*mining for patterns* in the plurality of entries in the second data structure to locate *multiples of a same thread id* that repeat.” The recited portions of Torii vaguely describe *searching* “information.” Even assuming, *arguendo*, that some portion of Torii teaches or suggests “mining for patterns,” the recited portion of Torii also does not teach or suggest “with respect to at least two of the plurality of groups of contiguous cache lines.” It is entirely unclear which part of Torii even teaches or suggests a “plurality of *groups* of contiguous lines.”

The Examiner provides a detailed argument for combining Torii and Steckermeier. However, the motivation provided can be found in neither reference, and thus, is purely speculative. The Examiner argues that “[i]t would have been obvious to one of ordinary skill in the art to combine Torii and Steckermeier since both consider the problem

of scheduling a thread on a processor such that the data the thread accesses is in the cache.” This reasoning suggests that the Examiner combined Torii and Steckermeier simply because they *can* be combined. This is clearly improper, and implies an improper hindsight reconstruction of the claims. The Examiner further asserts that: “Clearly, some mechanism is required to represent the thread’s cache locality in the system in addition to requiring some method of identifying the locality. Steckermeier’s silence on how to achieve these aspects of the scheduling essentially leave the means of achieving this up to the system engineering.” Once again, the Examiner’s own line of reasoning amounts to improper hindsight reconstruction, rather than showing a proper motivation or suggestion in the references themselves to be combined in the manner argued by the Examiner.

With regard to claim 16, Applicants maintain that § 3.2 of Steckermeier does not teach or suggest “*mapping* the threads identified by the located multiples of the same thread id to at least one native thread.” The Examiner argues that “[i]t is respectfully submitted that the scheduling of threads on a processor *inherently* meets this limitation.” The Examiner apparently is introducing an *inherency* argument. It is respectfully submitted that the Examiner has not met the requirements for *prima facie* inherency under MPEP 2112. In particular, the Examiner’s *conclusory* statement the scheduling of threads on a processor inherently meets the step of mapping, as claimed in claim 16, is insufficient. “In relying upon the theory of inherency, the examiner must provide *a basis in fact and/or technical reasoning* to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art.” Ex parte Levy, 17 USPQ2d 1461, 1464 (Bd. Pat. App. & Inter. 1990).

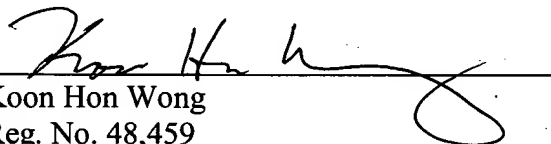
With regard to claim 33, Applicants maintain that § 3.2.2 of Steckermeier does not teach or suggest “*identifying pools of threads* in the plurality of entries in the second data structure such that each of the pools of threads comprises the threads identified by a same thread id that forms a multiple with respect to one of the plurality of groups of contiguous cache lines, the multiple repeating with respect to at least two of the plurality of groups of contiguous cache lines.” The arguments provided above for claim 1 apply similarly to claim 33.

Accordingly, independent claims 1, 16 and 33 are believed to be patentably distinguishable over the combination of Torii and Steckermeier. Dependent claims 2-15 and 17-32 are believed to be allowable for at least the reasons given for claims 1 and 16. Withdrawal of the rejection of claims 1-33 is respectfully requested.

In view of the foregoing remarks, it is respectfully submitted that all the claims now pending in the application are in condition for allowance. Early and favorable reconsideration is respectfully requested.

Respectfully submitted,

By:


Koon Hon Wong
Reg. No. 48,459
Attorney for Applicants

F. CHAU & ASSOCIATES, LLC
130 Woodbury Road
Woodbury, NY 11797
Telephone: (516) 692-8888
Facsimile: (516) 692-8889